

CS100M: Lab 14 Exercises for May 2, 3

In today's set of exercises, you will practice working with two-dimensional arrays in Java. You will implement several methods in the class `MatrixFun`. Each of these methods is **static**, and the class has no constructors or instance variables—for today, you can blissfully forget about writing object-oriented code. Instead, we will focus on performing operations on matrix-like arrays in Java. To begin, download the file `MatrixFun.java` from the course web site.

1. Implement the method `largestColumnFirst()`. This method takes a two-dimensional array of integers M , finds the column with the largest sum, and switches it with the first column in M . You can assume that M represents a rectangular matrix (i.e. it is not ragged).

For example:

6	7	9	4	8		9	7	6	4	8
3	2	7	4	1	$\Rightarrow$	7	2	3	4	1
9	4	5	8	3		5	4	9	8	3

Hint: Once you find the largest-sum column, you will need to swap its entries with the first column, element-by-element. Notice that this is different from the way rows can be swapped, as was shown in lecture. Can you see why?

2. Implement the method `transpose()`. This method takes a two-dimensional array of integers M , representing a square $n \times n$ matrix, and swaps the row and column of each element in M (i.e. reflects the contents of M over the main diagonal).

For example:

6	7	8	0		6	3	1	2
3	2	4	5	$\Rightarrow$	7	2	5	0
1	5	8	2		8	4	8	9
2	0	9	3		0	5	2	3

3. A two-dimensional array with n rows is said to be a *lower triangular matrix* if each row k has exactly k columns, for $k = 1 \dots n$. Implement the method `triToSym()` which takes a two-dimensional array T of integers as a parameter. If T is a lower triangular matrix, `triToSym()` returns a new square symmetric matrix with elements of T reflected above the main diagonal. Otherwise, the method returns **null**.

For example:

6					6	3	1	2
3	2				3	2	5	0
1	5	8			1	5	8	9
2	0	9	3		2	0	9	3

RETURNS

6	7	8	0					
3	2	4	5					
1	5	8	2					
2	0	9	3					

RETURNS

null

Challenge Exercise: Implement the method `raggedToSquare()`, which takes as a parameter a ragged two-dimensional array R of integers. Let r be the number of rows in R , c be the maximum number of columns in any row, and $n = \max(r, c)$. Your method should return a new $n \times n$ square matrix which encompasses the original ragged 2-D array R , and fills the matrix coordinates not present in R with random integers ranging between the minimum and the maximum values in R .

For example:

-3	7			
4	2	1	-7	8
-1	5	-4		
9				

RETURNS

-3	7	-5	0	3
4	2	1	-7	8
-1	5	-4	7	-2
9	-6	-4	8	-1
-1	4	9	-3	2

In the above example, $r = 4$, $c = 5$, and the minimum and maximum values are **-7** and **9**, respectively. In the returned matrix, the slanted numbers represent entries randomly added to make the matrix square. Note that an entire new row needed to be added, since $c > r$.

Finally, note that if the matrix R is square, then your method should return a new matrix that is an identical copy of R .

Please delete your files from the computer before you leave!